Rui Wang, Leonardo Gama

ECE 2804

Prof. Han

12/1/2020

Final report

Table of content

Introduction

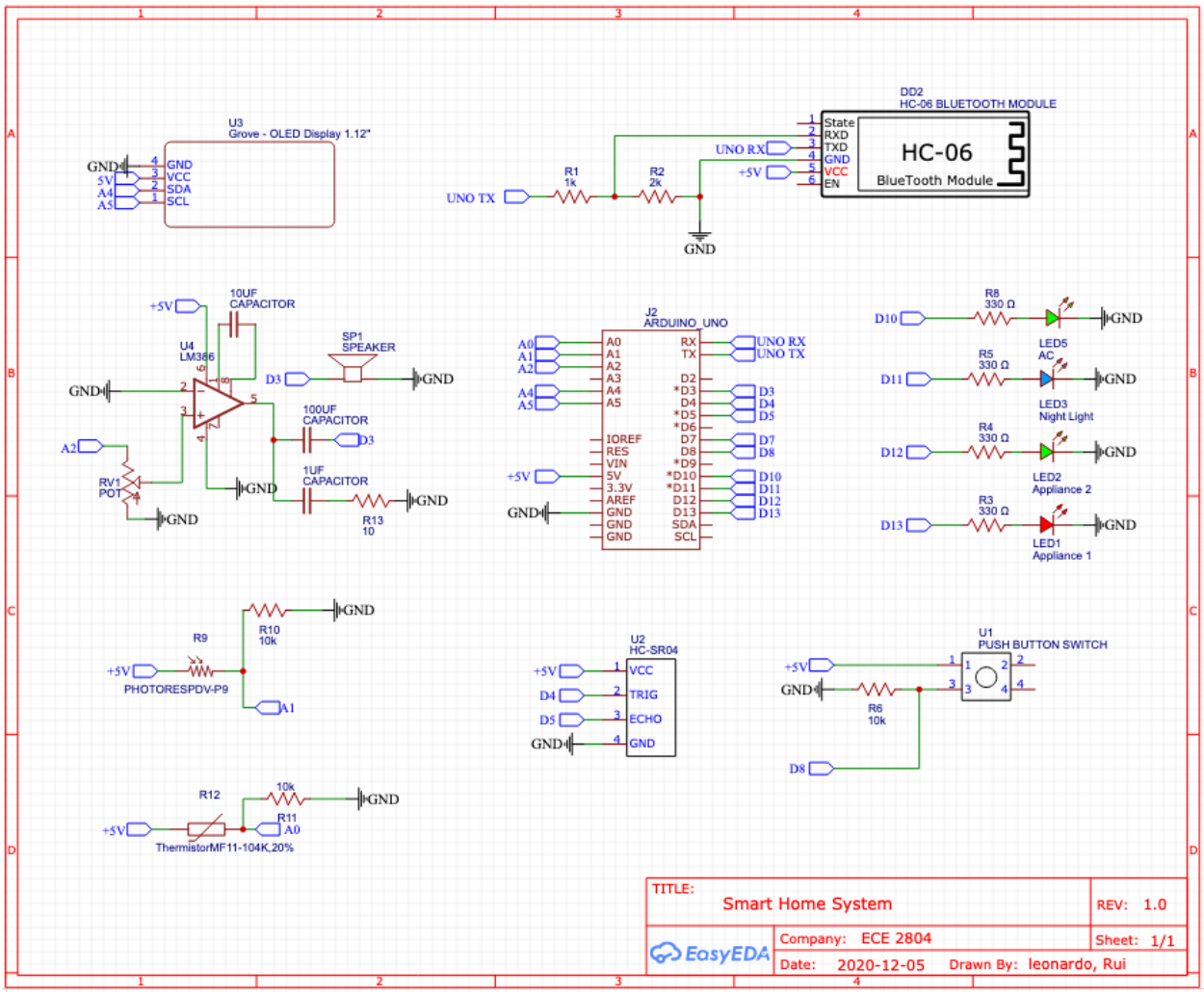
This report summarizes the work we have done for this project during this semester. They are 9 minimum subsystems which are required to do :

- OLED display can show temperature and humidity
- An audio amplifier can deliver message to alert people temperature is high
- A weather station measures temperature and humidity
- A night light turns on when it gets dark
- An appliance turns on when the temperature is reaching a certain threshold

We had as main goal in this project, implementing all required subsystems as optimals as we could. Using the necessary hardware and connections, always looking for the most simple but efficient way. This approach is also present in the android application, where we aimed for a minimal design without clutter or to extra flare, making it simple but efficient and easy to use.

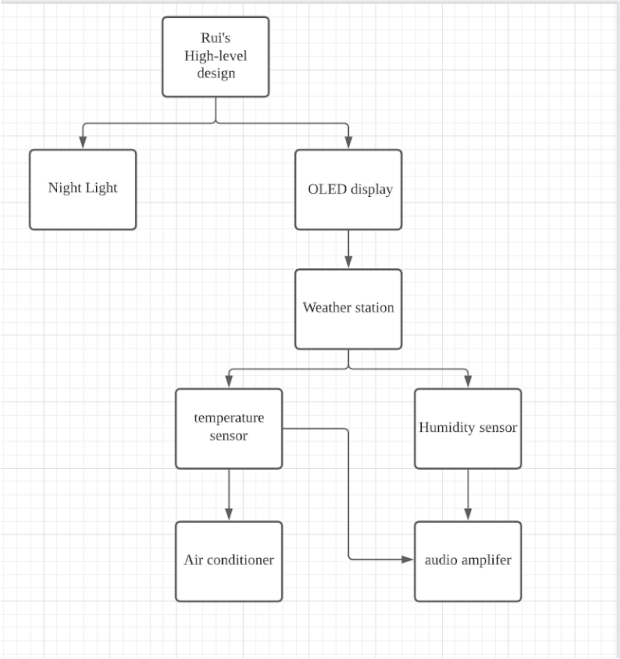
High Level Design

In the beginning of the project, we separated the work into two parts: implementation of MIT inventor and the hardware and software design of the circuit. Leo was in charge of the former, and Rui was responsible for the latter. The reason why we did this way was because the bluetooth module and MIT inventor were not related to the OLED display in the first place. Rui shared the hardware design and software design of the weather station with Leo, so both of us can work on the project simultaneously. What we have done individually is on the authorship page. Then at the end of the semester, we integrated our work together, including sharing code, simulation, and combining the schematic of the circuit into one picture. (Note: The schematic of humidity sensor is not in it, because there is no HS1101 schematic in the inventory of the website)



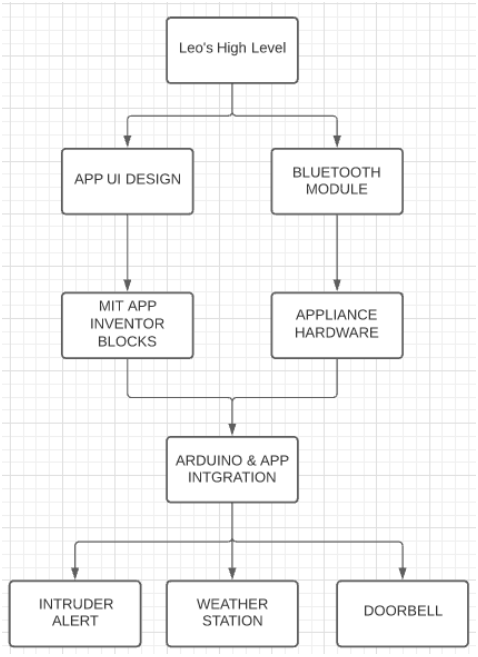
Notice that there are some connections here that are different from what we have in the detail design section, because in the detailed design, we had our circuit separately, and the schematic here is what we have for the final project. In addition to that, the Arduino code we are going to submit is following this schematic.

Rui’s high-level design



Here is a flowchart of steps on how I implemented my parts of the project. The reason I started with OLED display and night light implementation is because the weather station requires using the OLED to test out whether the values are correct or not. Since there are multiple subsystems we need to cover, it’s better for us to test each of them immediately after we finish one. Moreover, both air conditioner and audio amplifier are depending on the temperature and humidity. The night light can implement simultaneously with the OLED display because it’s only depending on the darkness of the photoresistor. The weather station contains a temperature sensor and humidity sensor. We did not test out after finishing both of them. Instead, we test out one at each time because if once one of them does not work properly, we can fix it right after we test it. Again, we think it is crucial to make sure that we implement every component correctly before we move on. Same logic applies to the night light implementation and audio amplifier since they are independent from each other.

Leo’s High-level design:

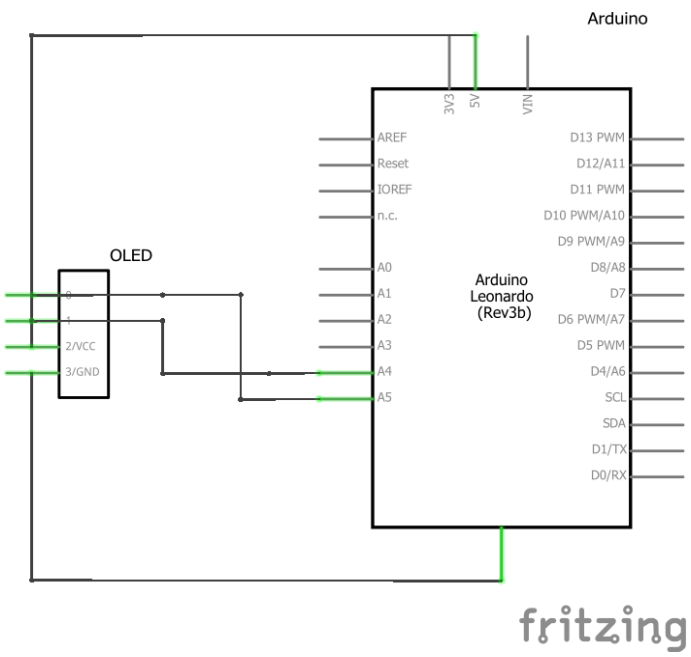


The decision and approach I took when starting my part on this project was to start by designing the most essential subsystem of this project, android app interface (UI) and Bluetooth module, since for most of the testing for other subsystems controlled or monitored by the android app would require a Bluetooth connection and user interface. After that I started focusing on the hardware and arduino code part for the LEDs representing appliances, as well the block part of MIT app inventor. With all of that, the integration of both parts and testing was done, at that point we already had a functional application that could communicate and control the arduino hardware via bluetooth. Having a basic version of the android application that could connect to bluetooth and control appliances, the implementation and integration of other subsystems were made much easier, since the “foundation” was already completed. Therefore I left for last the implementation and integration of intruder alert, weather station and doorbell, concluding and testing each one separately for easy troubleshooting.

Detailed Design

1. OLED display

The goal of this subsystem is to display temperature and humidity value on the OLED. The schematic of the hardware design is provided below.



There are four labels on the OLED display: GND, VCC, SCL, SDA. The VCC is connected to 5v, and GND is connected to GND on Arduino. SDA stands for Data which is connected to A4, and SCL stands for Clock, which is connected to A5

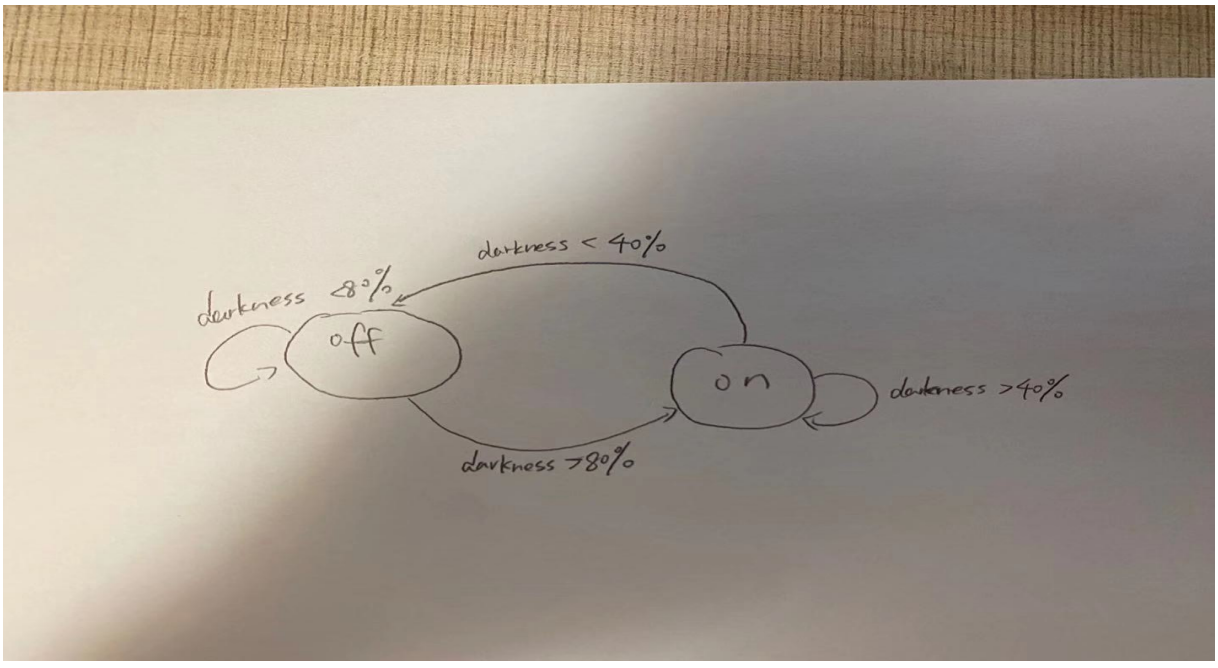
For the Arduino coding, I used a library called Ada fruit, which provides a structure for users to display characters on the OLED board.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

```
display.clearDisplay(); //clear the screen
display.setTextSize(1); //set the text size to be 1
display.setTextColor(WHITE); //set the color of the text to be white
display.setCursor(0,0); //set coordiniate of my text at (0,0)
display.println("TEM(C):   "); //display "TEM(C):"
display.println(T); //display temperature
display.println("HUM(%):   "); //display "HUM(%):"
display.println(humidity); //display humidity
display.display();
```

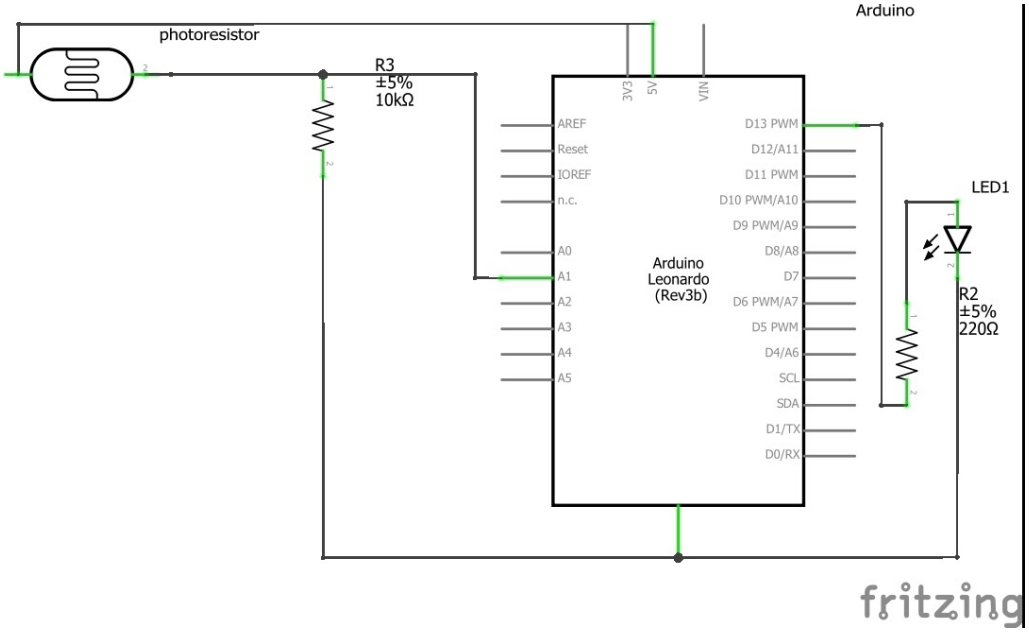
2. An automated night light

For this subsystem, we are required to achieve that when it is dark, the light should be automatically turned on, and when the environment is light, the LED should be automatically turned off. In addition, we are also asked to apply hysteresis on it. In our case, we set that the light turns on when darkness is more than 80%, and turns off when it's lower than 40%. A FSM is provided below.



For the hardware design, a photoresistor is connected to 5v and in series with a 10k resistor, and also connected to A1 in Arduino. The LED is in series with a 220 ohms resistor

connected to pin 13 on Arduino. A1 is used to measure the strength of darkness by the resistance of the photoresistor, and Pin 13 is used to control the light. The 10k resistor is used to prevent the circuit becoming a short circuit when it's fully dark, and the resistance of the photoresistor becomes zero.



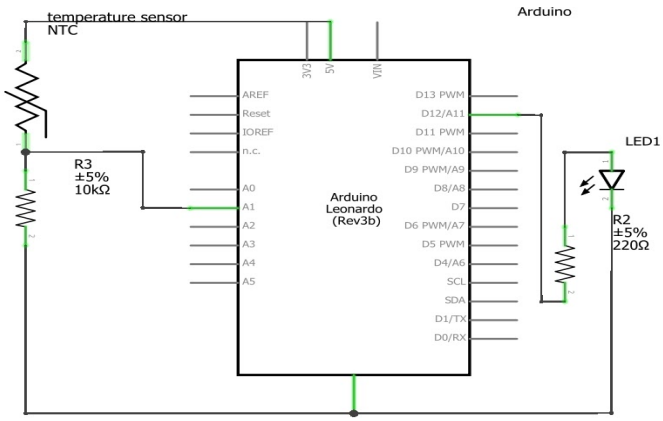
Moreover, according to the FSM, there are only two states (on and off), so Rui chose to use a boolean equation to separate the two conditions. The simulation is in this video:https://youtu.be/k7lurblNOh4.

```
const int Ldr = A1; //the themistor is connected to A1
```

```
int ldrstatus = analogRead(Ldr);
bool ledstatus = true; //initilize that ledstatus = true
  if(ledstatus == true){ //when ledstatus = true
    if(ldrstatus <= 300){    // darkness > 80%
    digitalWrite(Led, HIGH);  //turn on the led
    bool ledstatus = false; //ledstatus = false
    }
}
  else if (ledstatus == false){// when ledstatus = false

    if(ldrstatus >= 600){ //if the darkness < 40
    digitalWrite(Led, LOW); //turn off the led
    bool ledstatus = true; //ledstatus = true
    }
  }
```

3. Weather station

The weather station subsystem consists of 2 parts: Temperature station and humidity station. The requirement of it is to measure both values properly, and display them on the OLED.



For the hardware design of the temperature station, it has a thermistor (10k ohms max) in series with a 10k resistor, and there is wire between them connected to pin A1 on the Arduino. The 10k resistor is used to prevent that if the temperature was low enough to cause the resistance of the thermistor to become 0, then the circuit would become a short circuit. The LED on the left side is used for the air conditioner, and it will be discussed in the AC subsystem.

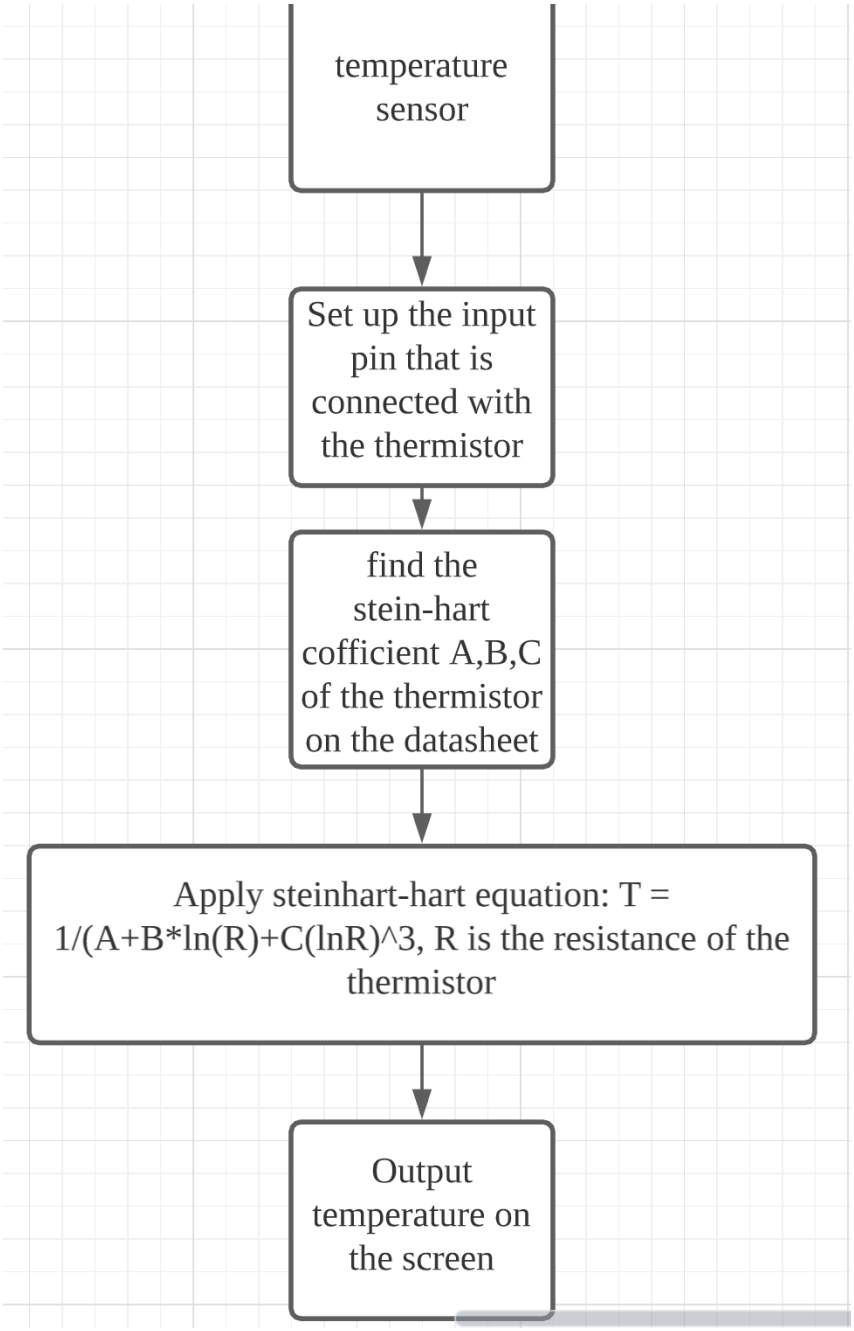
The equation is used to calculate the temperature is called Steinhart-Hart equation.https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation.

$$\frac{1}{T} = A + B\ln R + C(\ln R)^3,$$

where

T is the temperature (in kelvins),
R is the resistance at T (in ohms),
A, B, and C are the **Steinhart–Hart coefficients**, which vary depending on the type and model of thermistor and the temperature range of interest.

In our case, R would be the resistance of the thermistor, which can be calculated by using the voltage divider equation since it's in series with a 10k resistor. And the value of A, B, C can be found in the datasheet of the thermistor.

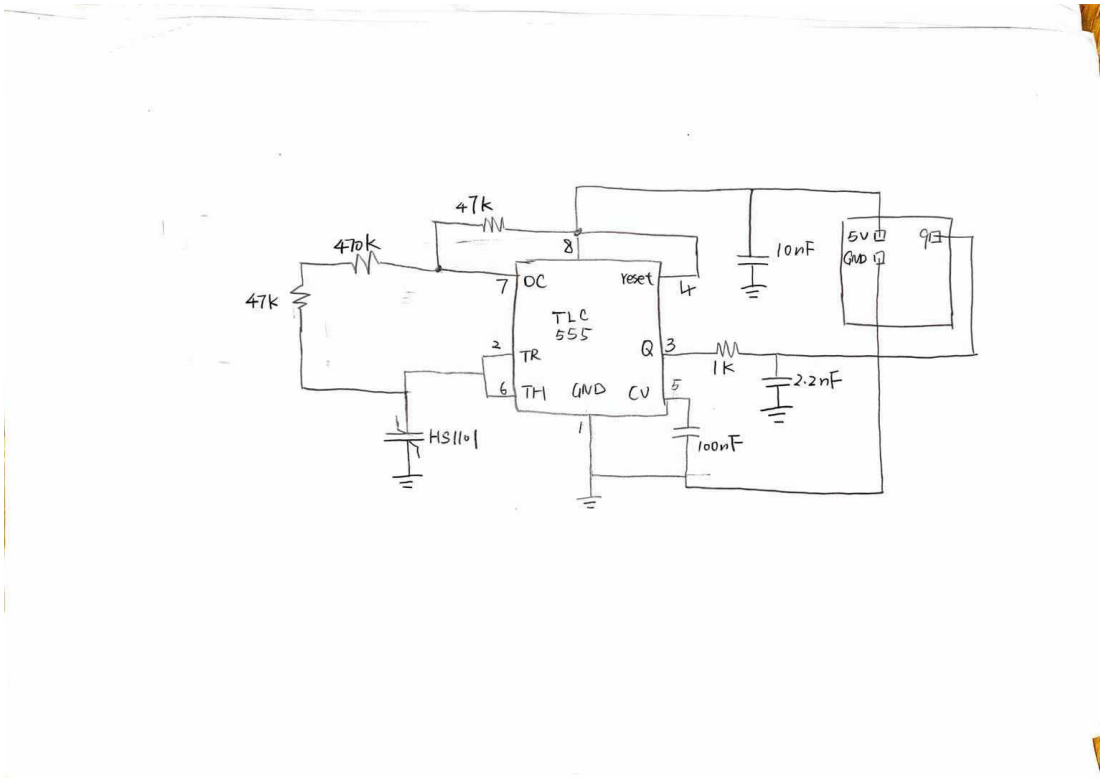


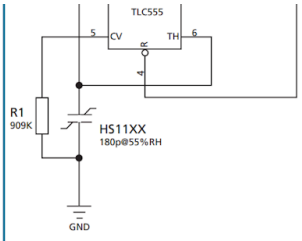

```
const int ThermistorPin = 0; //int Vo, and the pin0 which was connected to temperature sensor
int Vo;
float R1 = 10000; //R1 is 10K
float logR2, R2, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07; //coefficient A,B,C in the steinhart formula
```

```
Vo = analogRead(ThermistorPin);  //read the voltage
R2 = R1 * (1023.0 / (float)Vo - 1.0); //calculation the resistance of themistor
logR2 = log(R2);
T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2)); //apply steinhart-Hart equation: T = 1/(A+BlnR+C(lnR)^3)
T = T - 273.15;   // convert to Celsius
```

```
display.println("TEM(C):   "); //display "TEM(C):"
display.println(T); //display temperature
```

For the hardware design of the humidity station, the schematic is shown below, it's based on the HS1101 datasheethttps://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FHPC052_J%7FA%7Fpdf%7FEnglish%7FENG_DS_HPC052_J_A.pdf%7FCAT-HSC0006. However, there is also a little difference between them. In the data sheet, there is a 50k potentiometer, but I used a 47k regular resistor instead. Because in our case, we don't need to use a potentiometer to change the resistance, we only need a constant resistor to measure the humidity properly.





BILL OF MATERIAL AVAILABLE ON REQUEST

Since the charge and discharge of the sensor run through different resistors, R2 and R4, the duty cycle is determined by :

$$t_{high} = C@\%RH*(R2+R4)*ln2$$
$$t_{low} = C@\%RH*R2*ln2$$
$$F = 1/(t_{high}+t_{low}) = 1/(C@\%RH*(R4+2*R2)*ln2)$$
$$Output\ duty\ cycle = t_{high}*F = R2/(R4+2*R2)$$

To provide an output duty cycle close to 50%, R4 should be very low compared to R2 but never under a minimum value.
Resistor R3 is a short circuit protection. 555 must be a CMOS version.

REMARK

R1 unbalances the internal temperature compensation scheme of the 555 in order to introduce a temperature coefficient that matches the HS1100/HS1101 temperature coefficient. In all cases, R1 should be a 1% resistor with a maximum of 100ppm coefficient temperature like all other R-C timer resistors. Since 555 internal temperature compensation changes from one trademark to one other, R1 value should be adapted to the specific chip. To keep the nominal frequency of 6660Hz at 55%RH, R2 also needs slight adjustment as shown in the table.

555 Type	R1	R2
TLC555 (Texas)	909kΩ	576kΩ
TS555 (STM)	100nF capacitor	523kΩ
7555 (Harris)	1732kΩ	549kΩ
LMC555 (National)	1238kΩ	562kΩ

For a frequency of 6660Hz at 55%RH

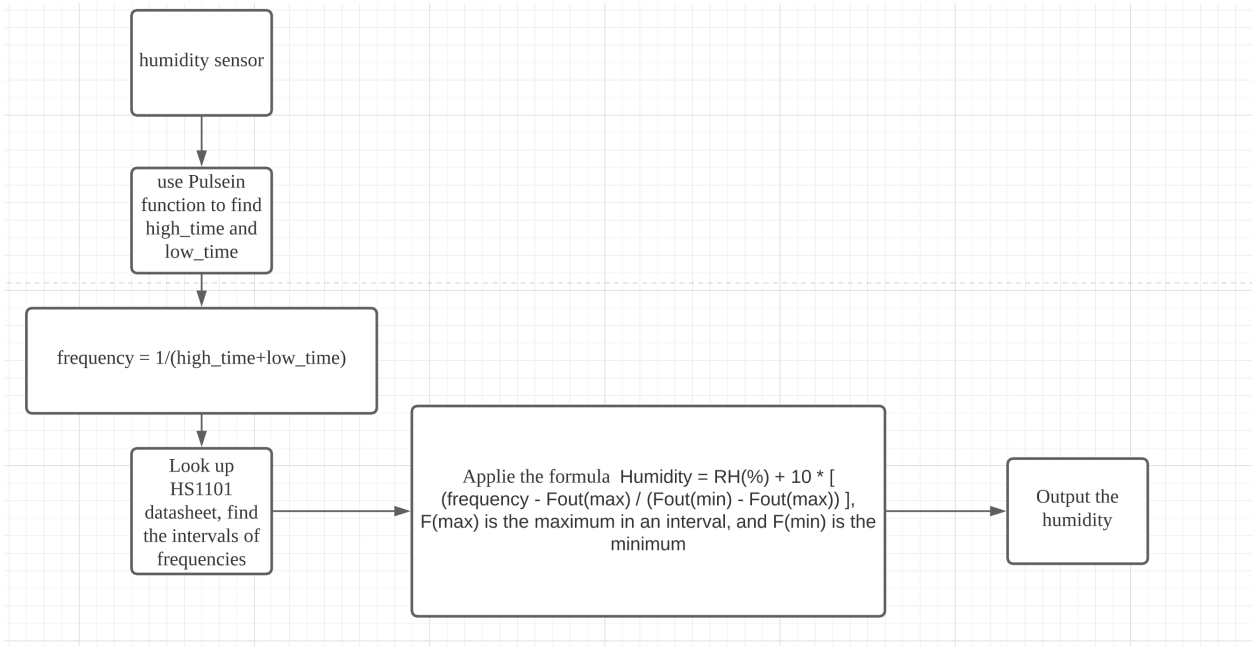
Typical Characteristics for Frequency Output Circuits

REFERENCE POINT AT 6660Hz FOR 55%RH / 25°C

RH	0	10	20	30	40	50	60	70	80	90	100
Frequency	7351	7224	7100	6976	6853	6728	6600	6468	6330	6186	6033

Typical for a 555 Cmos type. TLC555 (RH : Relative Humidity in %, F : Frequency in Hz)

According to the datasheet, in order to develop the code for calculating the humidity, we need to find " high time" and "low time" first by using the "pulse in" function in Arduino. Then use them to find the frequency. Moreover, the calculation of humidity is followed by this equation: humidity = RH(%) + 10 * [(frequency - Fout(max) / (Fout(min) - Fout(max))]. For instance, if the frequency = 6188, then the current humidity would be equal to 80 + 10*(6188-6330)/(6188-6330). The software code with comments is provided below.




```
const int input = 9; // RC circuit with HS1101 sesnor connected to digital pin D9
int high_time; // initial the high and low time, float period, humidity, and frequency since they are variables.
int low_time;
float humidity;
float frequency;
```

```
  high_time=pulseIn(input,HIGH); //t_high
  low_time=pulseIn(input,LOW); t_low
  frequency=1000/(high_time+low_time); //calcaute the frequency
```

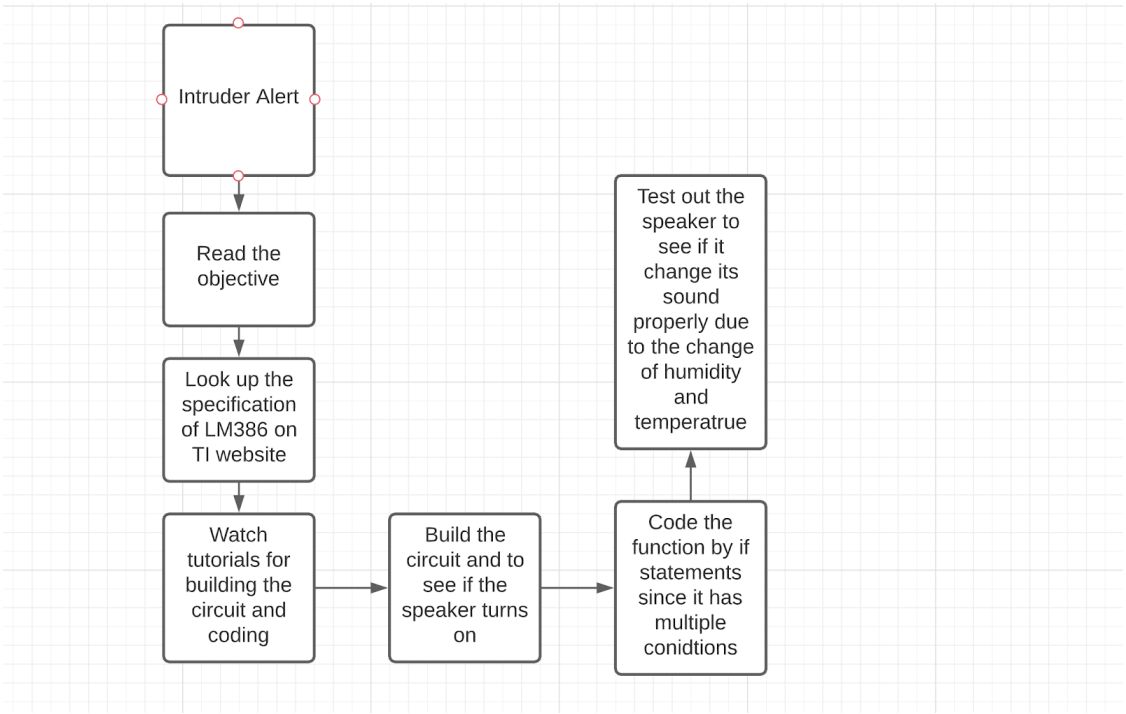
```
  //a general formula for humidity:
  //Humidity = RH(%) + 10 * [ (frequency - Fout(max) / (Fout(min) - Fout(max)) ]
 if (frequency <= 6186 && frequency >= 6033) //when the frequency is between 6186 and 6033, RH=90
    {
    humidity = 90 + 10*((frequency-6186)/(6033-6186));
    }
  else if(frequency <= 6330 && frequency > 6186) //when the frequency is between 6330 and 6186, RH = 80
    {
    humidity = 80 + 10*((frequency-6330)/(6186-6330));
    }
  else if(frequency <= 6468 && frequency > 6330) //when the frequency is between 6468 and 6330, RH=70
    {
    humidity = 70 + 10*((frequency-6468)/(6300-6468));
    }
  else if(frequency <= 6600 && frequency > 6468) //when the frequency is between 6600 and 6468, RH=60
    {
    humidity = 60 + 10*((frequency-6600)/(6468-6600));
    }
  else if (frequency <= 6728 && frequency > 6600) //when the frequency is between 6728 and 6600, RH=50
    {
     humidity = 50 + 10*((frequency-6728)/(6600-6728));
    }
  else if(frequency <= 6853 && frequency > 6728) //when the frequency is between 6853 and 6728, RH=40
    {
    humidity = 40 + 10*((frequency-6853)/(6728-6853));
    }
  else if(frequency <= 6976 && frequency > 6853) //when the frequency is between 6976 and 6853, RH=30
    {
    humidity = 30 + 10*((frequency-6976)/(6853-6976));
    }
  else if(frequency <= 7100 && frequency > 6976)//when the frequency is between 7100 and 6976, RH=20
    {
    humidity = 20 + 10*((frequency-7100)/(6976-7100));
```

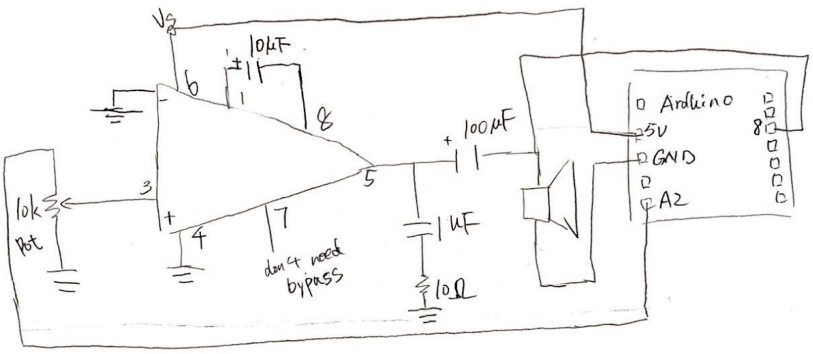
```
  }
else if(frequency <= 7224 && frequency > 7100) //when the frequency is between 7224 and 7100, RH=10
  {
  humidity = 10 + 10*((frequency-7224)/(7100-7224));
  }
else if (frequency<= 7351 && frequency >= 7224)  //when the frequency is between 7351 and 7224, RH=0
  {
    humidity = 10*((frequency-7351)/(7224-7351));
  }
```

The testing video:https://www.youtube.com/watch?v=hlmSr9GzIJ8

4. An audio amplifier

The requirement for this subsystem is to design an alert which turns on if the temperature is high.





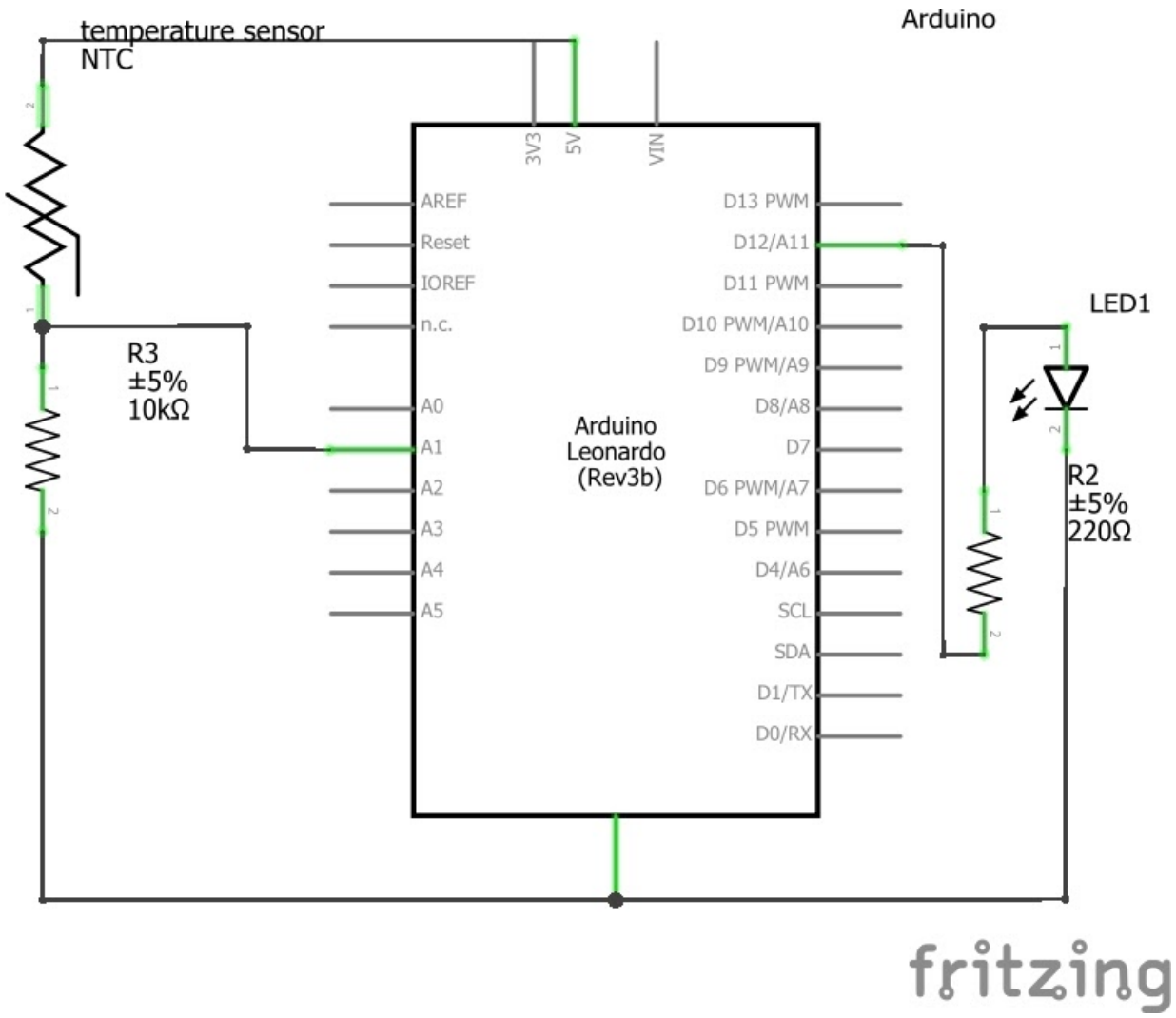
The schematic diagram is based on the audio amplifier on TI website. However, we didn't have a 250 uF capacitor and it wasn't necessary to use more than 200 uF since we only needed the speaker to have a warning sound. Hence, we used 100 uF instead. Moreover, we also have the speaker connected to the output pin 8 on Arduino to make it work. The 10k potentiometer also plays an important role: it connects with Vin on LM386 and the A2 on Arduino as an input for the system.

The audio amplifier gives a warning when the temperature and humidity reach a certain threshold that humans are uncomfortable with. The fundamental component we used for this system is LM386 known as an audio amplifier. The website we referred while building the circuit was https://www.ti.com/lit/ds/symlink/lm386.pdf . The alert is for seniors who can't suffer either high temperature or humidity. When the alert is up, they realize they need to turn on the air conidtioner. The code is provided below. Basically the sound can be changed by frequency depending on different situations.

```
int pin = 8; // read output pin 8

 if (humidity > 60 && T > 30){ //the alert turns on when Temperature > 30 C and
humidity > 50%
tone(8,1136,2000);//a tone(Pin, frequency, duration) 50ms duration.
 }
 else if (humidity > 60){
noTone(8);//b //notone is to turn off the previous sound.
tone(8,1915,2000);//c
 }
 else if (T > 30 ){
  noTone(8);//b
  tone(8,1700,2000);//c
 }
 else if (T <= 30 && humidity <= 60){
  noTone(8); //if no value is reaching the threshold, turn off the speaker.
 }
```

5. **An appliance turns on when the temperature is reaching a certain threshold (Air conditioner)**

The design of this subsystem is related to the weather station. We have a LED which turns on when the temperature is higher than 35 C, and turns off when it's below 33 C.

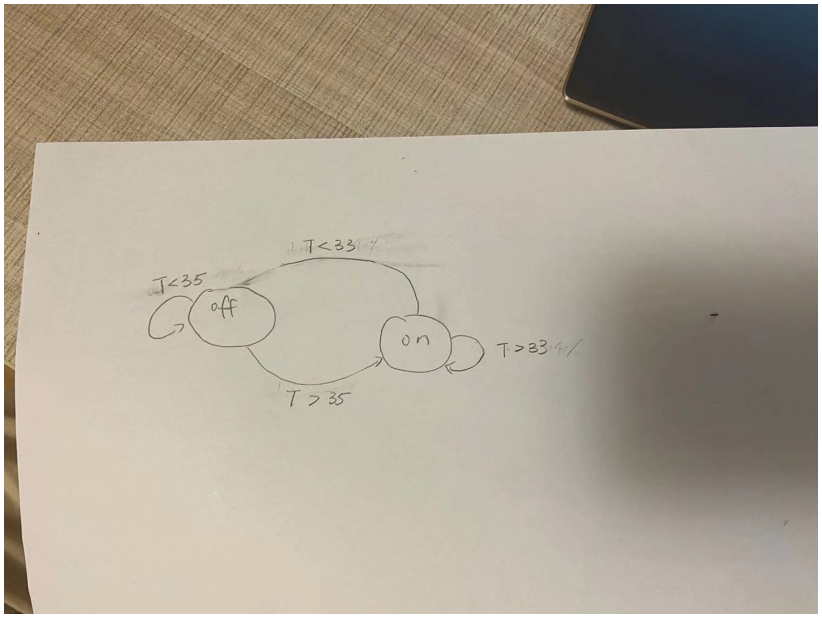


For the hardware design, there is a LED in series with a 220 ohms connected to Pin 12 on the Arduino board.

For the software design, we came up with a FSM which shows how it works.

If the temperature is greater than 35 C, the light turns on. Otherwise it remains off. When the light is on, if the temperature is below 33 C , then the light turns off. Otherwise it remains on. The testing video and code is provided:https://www.youtube.com/watch?v=WyeBf67mUW0

```
const int AC = 12; //the led is connected to outpin 12 on arduino
```



MIT APP Inventor

To meet this requirement, we had to develop an APP using MIT APP Inventor, that is capable of controlling and monitoring all the appliances and the weather station from the Smart home system. The app has 3 different screens in the app:

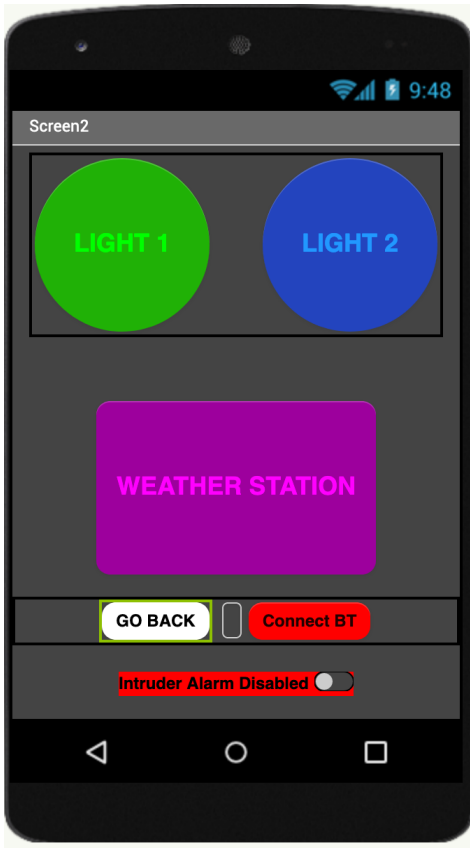
Screen 1: Home Screen

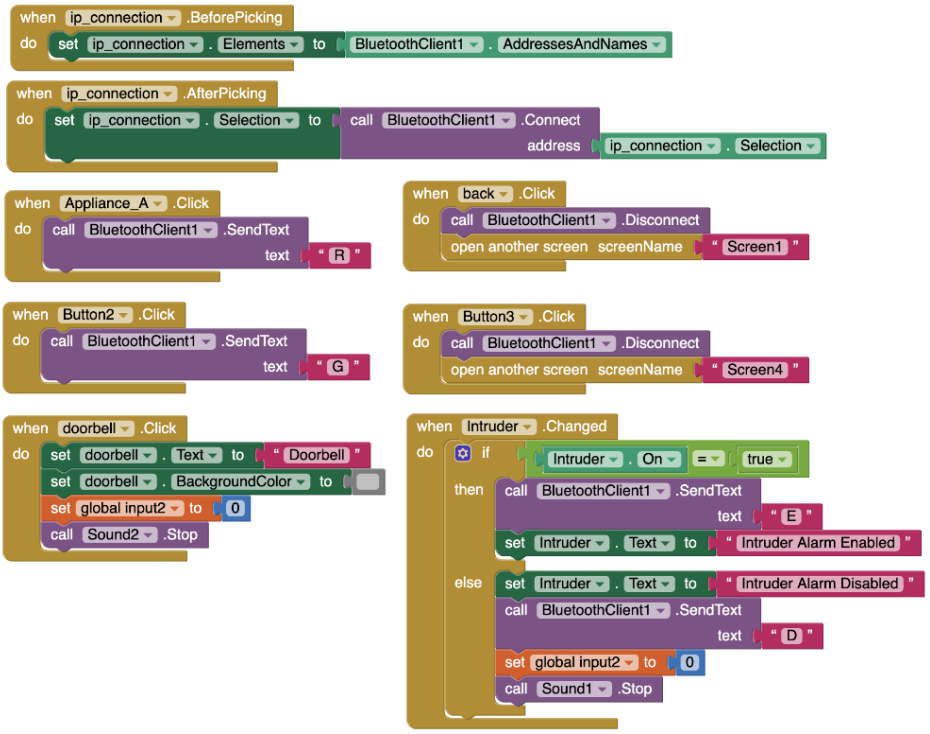
In the home screen the user is welcomed with the name of the app, and is able to press one button that will change the app to another screen.

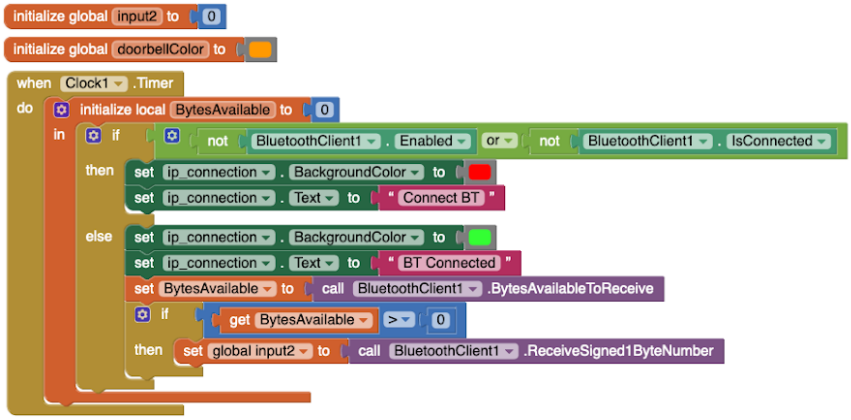
By pressing "**USE APP" = Button2**, according to the blocks, it will direct the user to **Screen2**.

Screen 2: Smart Home Monitoring and Control

In this screen, the user is able to control Appliances, monitor the weather station, and is also able to enable/disable and receive Intruder Alert, doorbell notifications and connect to the bluetooth module.







By pressing **“LIGHT 1”** the app sends text **“R”** to the arduino which is coded to turn on or off the red LED representing Appliance A, whenever it receives the character **“R”** (check Bluetooth Module section for Arduino code).

Same thing happens for **“LIGHT 2”**, but instead of sending text **“R”** it sends **“G”** (check blocks). The arduino section of code responsible for this can be found bellow.

```
// Function responsible for the communication and intepretation of data to be sent and receveid through arduino bluetooth module
// and MIT app inventor app.
void ApplianceControl() {

  //Turns Red LED on or off if data received is "1"
  if (character == 'R') {
    //if state currently off, sets led to HIGH ==> turns led on
    if (redVoice == "off") {
      digitalWrite(redLed, HIGH);
      redVoice = "on";
      character = 'N';
    }
    //if state currently on, sets led to LOW ==> turns led off
    else if (redVoice == "on") {
      digitalWrite(redLed, LOW);
      redVoice =  "off";
      character = 'N';
    }
  }

  //Turns Green LED on or off if data received is "2"
  if (character == 'G') {
    //if state currently off, sets led to HIGH ==> turns led on
    if (greenVoice == "off") {
      digitalWrite(greenLed, HIGH);
      greenVoice = "on";
      character = 'N';
    }
    //if state currently on, sets led to LOW ==> turns led off
    else if (greenVoice == "on") {
      digitalWrite(greenLed, LOW);
      greenVoice = "off";
      character = 'N';
    }
  }
}
```

By pressing “**GO BACK”**, according to the blocks, it will direct the user back to **Screen1: Smart Home Monitoring and Control**.

By pressing “**WEATHER STAT.”**, according to the blocks, it will direct the user to **Screen4.**

The **“Connect BT”** button is coded with blocks to be smart, and when there is no bluetooth module connected, the button will turn red and change its text to **“Connect BT”**. After

pressing the button and selecting the Bluetooth module the device will be connected, the button will turn green and change its text to **“BT Connected”**.

This screen also includes a switch “**Intruder Alert**” which is used to enable or disable the alarm system for detecting intruders. When switch is off, alarm will not ring, when switch is on, if intruder is detected alarm will ring. This was implemented, because sometimes people have guests coming into their homes, and no one wants an alarm ringing loud when guests are entering the home. Taking a look at the blocks, all text received through bluetooth module is interpreted and stored in **global input2,** when global input equal to **ASCII 50 = decimal 2,** the alarm system will turn on and ring, notifying intruder through message on screen and sound. The arduino section of code responsible for this can be found below.

```
void IntruderAlert() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  cm = (duration/2) / 29.1;     // Divide by 29.1 or multiply by 0.0343
  inches = (duration/2) / 74;   // Divide by 74 or multiply by 0.0135

  // If data received is character 'E' enable ultrasonic sensor to read and inpret if there is intruder
  if (character == 'E') {
    intruder_enable = 0;
    character = 'N';
  }
  // If data received is character 'D' disable ultrasonic sensor to read and inpret if there is intruder
  else if (character == 'D') {
    intruder_enable = 1;
    character = 'N';
  }

  // If enabled, and if intruders is detected within range, send intruder = 2 thats is equivalent to ASCII 50
  if (intruder_enable == 0) {
    if (inches <= 10 && inches >=1) {
      Serial.println(intruder);
      delay(500);
      character = 'N';
    }
  }
}
```

Similarly, there is also a doorbell button (shown in the demo video) which is used to dismiss the doorbell when someone rings the doorbell by pressing a push button. When global input is equal to **ASCII 49 = decimal 1,** means there is someone ringing the doorbell, notifying that someone is at the door through a message on screen and sound. The arduino section of code responsible for this can be found below.

```
void DoorBell() {
  // Dorbell send notification when button presed and released to android app (PUSH BUTTON DEBOUCEND)
  currentState = digitalRead(button);

  // If push button is pressed and released, send doorbell = 1 thats is equivalent to ASCII 49
  if (lastState == HIGH && currentState == LOW) {
    Serial.println(doorbell);
    character = 'N';
  }

  lastState = currentState;
}
```

Screens 3: Weather Station

Screens 3 has the information collected by the weather station, that includes the humidity and temperature, both are updated in real time and match the information displayed on the OLED display.

Similarly to **Screen 2**, you have the "**Connect BT"** button, right after bluetooth connection is made, temperature and humidity reading will be displayed, replacing the black text "Temp Humidity". This is done by the blocks, after bluetooth is connected the android app will send every clock cycle the character "W" to arduino via bluetooth module. The arduino will interpret the character and send back the weather station temperature and humidity information. The snippet of code responsible for that is to be found below.

```
void WeatherApp() {
  // Weather Station Send Information to android app
  if (character == 'W') {
    myT = String(T); //convert temperature to string and store in myT
    myT = myT + "°C"; // add units to temperature
    myHumidity = String(humidity); //convert humidity to string and store in myHumidity
    myHumidity = myHumidity + "%"; // add units to humidity
    weather = myT + "   " + myHumidity; //cocatenate
    Serial.println(weather); // sends back weather to app through serial
    character ='N';
  }
}
```

By pressing "**GO BACK"**, according to the blocks, it will direct the user back to home screen

Screen2: Home Screen.

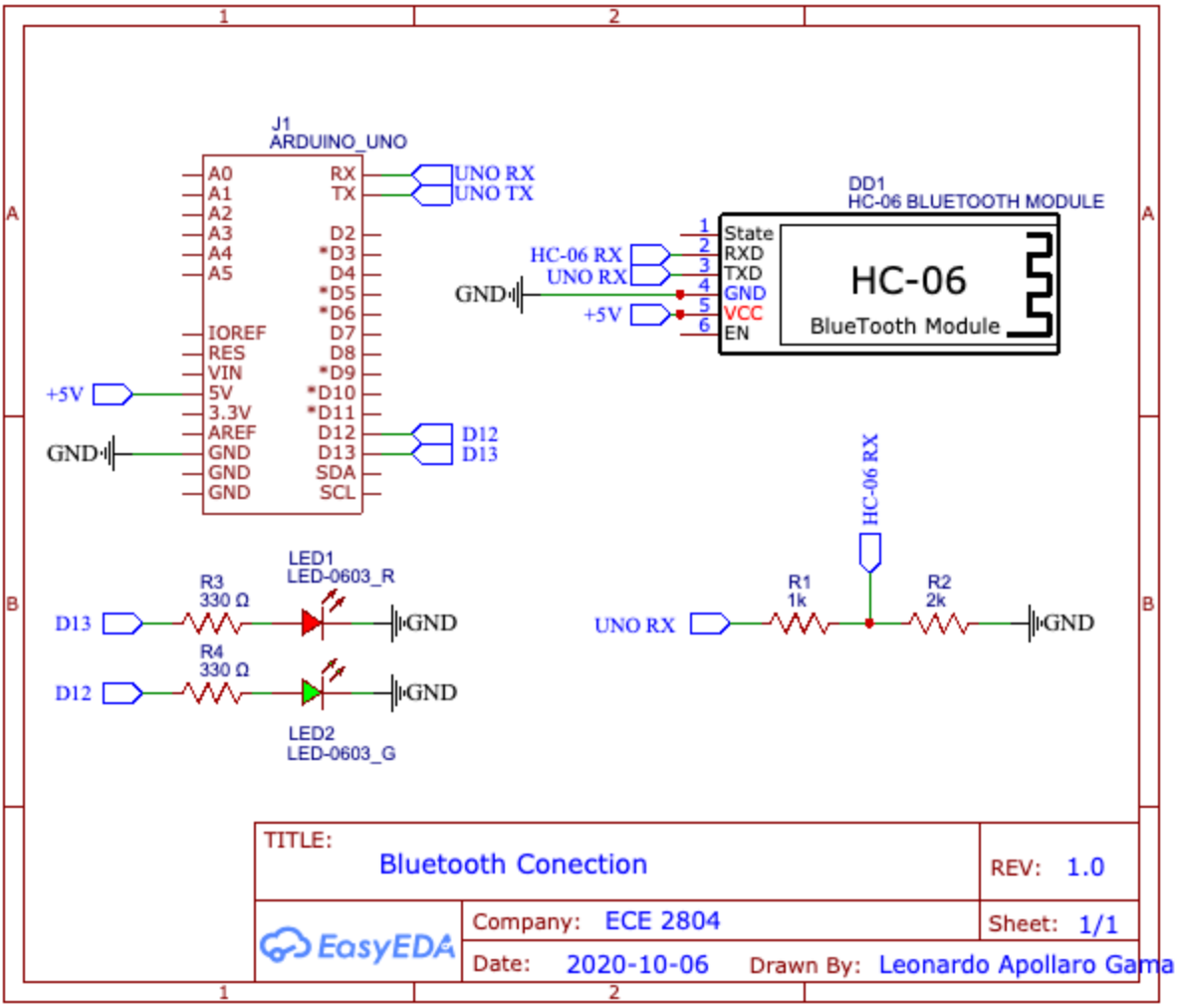
Demo video showing all app features, working and tested can be found in the link below:

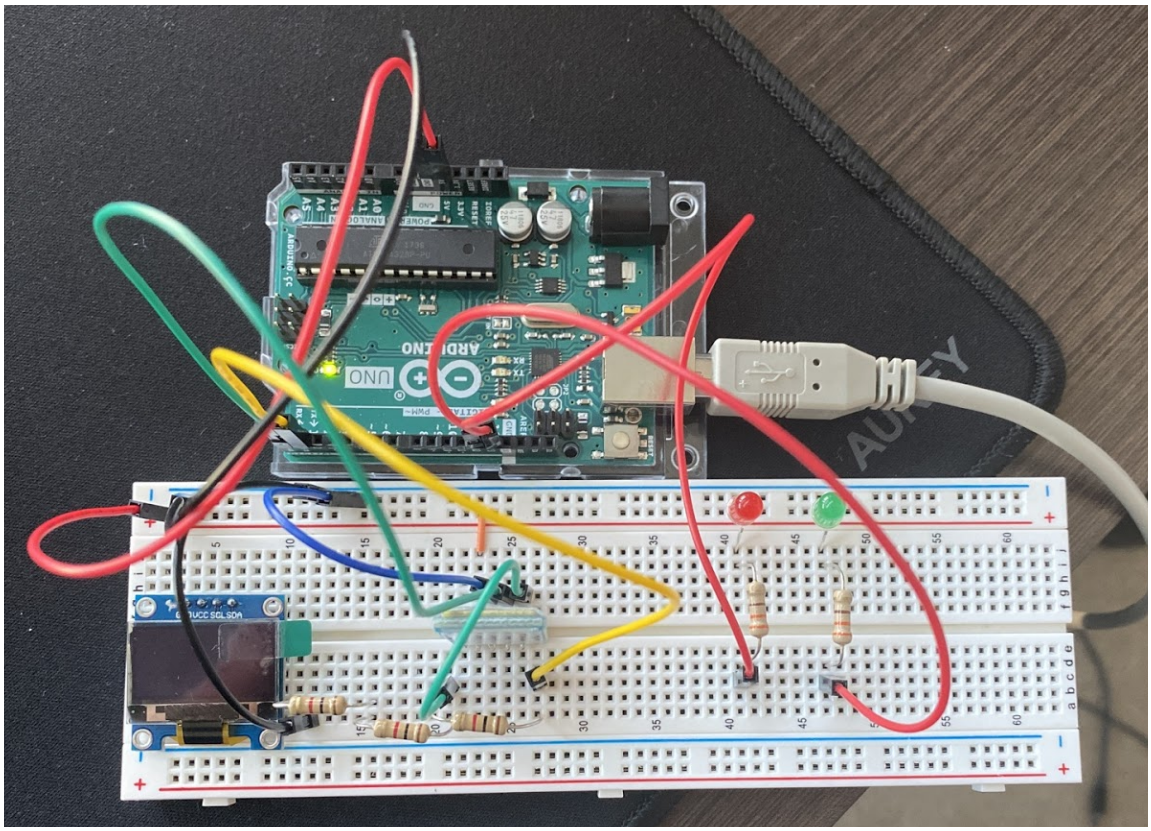
https://drive.google.com/file/d/127PM-4FcoVbnp8_BwNFYSNwElvHAhnPh/view?usp=sharing

Bluetooth Module

This requirement was met by writing an arduino code and making the right connections between arduino uno and the bluetooth module.

Below one is able to find the schematics of the circuit used to meet this requirement and a picture of the circuit.




ARDUINO
UNO
AUKEY

Validation of Overall Project

We have accomplished all of the basic requirements, and came up with the doorbell as an extra. Here are two videos that show validation of the project.

https://youtu.be/z0cxSf1hDhc

https://drive.google.com/file/d/127PM-4FcoVbnp8_BwNFYSNwElvHAhnPh/view?usp=sharing

Conclusion

This project gave us the opportunity to combine the knowledge that we learnt from other ECE courses. It was also the first time we teamed up with another ECE major student to work on a major-related project. Overall, we are really satisfied, proud and happy about the end product we accomplished with this project. In addition, we think we did well in this project in terms of communication, compromise, cooperation, and self-learning. We shared information about the project, kept updating the progress, and did the presentation and reports together. Moreover, the smart home project itself is practical since all of the subsystems can be used in daily life. It makes me feel satisfied when I successfully finish a subsystem and see it working. If we could do this project again, we believe meeting in-person and doing the work together instead of using zoom and discord would make us deliver a better design and system at the end. First of all, the hardware design could be much easier since it's difficult and not clear to show how a circuit is built, via the internet. Secondly, communication would go more smoothly. Not contacting people face to face seems like there is always a barrier. It would affect the efficiency of working, and the relationship between us. To summarize, it is a valuable engineering experience to have in college, and brings advantages to our future research and work.

Authorship page

Rui’s accomplishments

- OLED display
- Weather station
- Audio amplifier
- Night light
- Air conditioner

Leo’s accomplishments

- Bluetooth Module
- Appliance Controlling
- MIT App Inventor
- Intruder Alert
- Doorbell (extra)